UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola d'Enginyeria de Barcelona Est

TREBALL FI DE GRAU

Grau en Enginyeria de l'Energia

MODELLING AND SIMULATION OF AUTONOMOUS VEHICLES IN A MULTI-LANE AND HETEROGENOUS TRAFFIC ROAD

Memòria i Annexos

Autor: Miranda Elias, Marta
Director: Dòria-Cerezo, Arnau

Convocatòria: Setembre 2019

Resum

Aquest treball de final de grau conté la modelització matemàtica i el control, utilitzant la plataforma *Matlab* i *Simulink*, de cotxes autònoms amb diferents característiques circulant per un circuit format per múltiples carrils. Com a cos principal destaquen les equacions modelitzades per permetre simular fenòmens d'interacció entre vehicles, sent no tant important els fenòmens en si, sinó la obtenció de dades per poder-los portar a terme. Tot i això també s'han implementat relacions de conducció entre vehicles com la velocitat de creuer, o canvis de carril. Aquest últims, de forma molt bàsica, per utilitzar com a eina en propers projectes centrats únicament en aquesta forma de interacció. Cal emfatitzar també, l'estudi teòric fet en aquest projecte sobre les Corbes de Bézier i la seva utilitat pel disseny de circuits, així com la programació d'una eina que dona informació sobre dades energètiques de qualsevol escenari de circulació realitzat amb una flota amb cotxes autònoms. Per últim, s'avaluen les solucions de diferents simulacions realitzades amb els models matemàtics creats i es comenta, tan l'ús que podrien tenir com les modificacions a les quals podrien estar subjectes per ser utilitzats com a eina en futurs estudis.

Resumen

Este trabajo de final de grado contiene la modelización matemática y el control, utilizando la plataforma *Matlab* y *Simulink*, de coches autónomos con diferentes características circulando por un circuito formado por múltiples carriles. Como cuerpo principal destacan las ecuaciones modelizadas para permitir simular fenómenos de interacción entre vehículos, siendo no tan importante los fenómenos en si mismos, sino la obtención de datos para poderlos llevar a cabo. Sin embargo, también se han implementado relaciones de conducción entre vehículos como la velocidad de crucero, o cambios de carril. Este últimos, de forma muy básica, para utilizar como herramienta en próximos proyectos centrados únicamente en esta forma de interacción. Hay que enfatizar también, el estudio teórico realizado en este proyecto sobre Curvas de Bézier y su utilidad para el diseño de circuitos, así como la programación de una herramienta que da información sobre datos energéticos de cualquier escenario de circulación realizado con una flota con coches autónomos. Por último, se evalúan soluciones de diferentes simulaciones realizadas con los modelos matemáticos creados y se comenta, tanto el uso que podrían tener, como las modificaciones a las que podrían estar sujetos para ser utilizados como herramienta en futuros estudios.

Abstract

This project contains the mathematical modeling and control using *Matlab* and *Simulink* of autonomous cars, with different characteristics, circulating through a circuit consisting of multiple lanes. As a main body, the modeled equations stand out for allowing vehicles' interaction simulations, not being those interactions themselves so important, but the data obtention to carry them out. However, it has been implemented driving relationships between vehicles too, such adaptive cruise control or lane changes. This last one, in a very simple model, to use it as a tool in future projects focused only on this type of interaction. It is also necessary to emphasize the theoretical study carried out in this project on Bézier Curves and its usefulness for circuit design, as well as the programming of a tool that gives information about energy data of any traffic scenario carried out with an autonomous cars fleet. Finally, it has been evaluated solutions of different simulations performed with the mathematical models created. The finality of that is to comment the use that they could have and the modifications that could be make to them, to make them useful as tools in future studies.

Acknowledgments

This project would not have been possible without the help of my tutor. Thank you very much Arnau for giving me the opportunity to learn so much and have had plenty of patience teaching me programming. Thank you for your cheers and for supporting me (especially the last few weeks) letting me practically leave in your office.

I also want to thank my family and friends for getting behind me, especially in the worst moments.

Glossary

Matlab: it is a high-level technical computing language an interactive environment for algorithm development, data analysis, numeric computations and data visualization.

Simulink: it is an environment for multidomain simulation and Model-Based Design for dynamic and embedded systems founded on Matlab. It provides an interactive graphical environment and a customizable set of block libraries with the capability of design, simulate, implement, and test a variety of time-varying systems, including communications, controls, signal processing, video processing, and image processing.

PI Controller: feedback control mechanism which calculates the error deviation between a wanted value and a measured value. The algorithm PI consist in two parameters: the proportional and the integral. The proportional depends on the actual errors, and the integral on the past errors.

Adaptive Cruise Control (ACC): automatic system which calculates the speed difference between the own vehicle and the vehicle in front and corrects speed to keep a safety distance.

Energy usage: total amount of energy used to perform a displacement. Its units can be kJ, kWh, kcal or other energy units.

Index

1. PREFACE

1.2. ORIGINAL PROJECT

Currently the Institut d'Organització i Control de Sistemes Industrials (IOC) dedicates a part to investigate about the mathematical modelling and control of autonomous vehicles, even having some robotic prototypes. This added to some articles as *"A first order sliding mode-based adaptive cruise controller"* or *"Simulations of a vehicle following a path using Bézier curves"*, has becoming the origin to make a project that creates a tool to provide new information to the IOC investigations.

1.3. MOTIVATION

During this century, road vehicles have changed a lot. Not only the parts that conform them has improve, but the materials, the fuel and a lot of new applications have been added to them to make them more efficient, secure and autonomous. Although all those changes, there is a thing that has not been replaced: the human driver. But this is about to change.

The traffic problems, accidents and the little efficiency that has human driving, have proven that how more autonomous is a car, more secure is and less energy wastes. That has impulse the technology to star building complete autonomous cars. As all the technological novelties, they have fails at firsts, so the simulations are a good tool to design them. Not only allow to know the performance of a vehicle with any security risk but, give the possibility to change their design without having to waste the money or the time that would involve building several vehicles prototypes for all the important changes, that applies to them, during the design.

1.4. PRIOR PROJECTS

To decide how to approach the objective of this project it has been taken as frame as reference other degrees final thesis. Those are:

Energy efficiency comparison between human drivers and adaptive cruise control system (Marc Fernandez, 2017): consists on creating a mathematical model that has been developed with the aim of studying how vehicles behave in different traffic scenarios, not only in fluidity terms but also in energetic consumption.

Simulació de la fluència del trànsit de vehicles comparant el comportament humà i l'impacte de vehicles autònoms (Jaume Cartro,2017): develops a program capable of modelling and simulating the traffic flows in non-urban roads, and study how traffic jams are created.

Influence of the Cooperative Adaptive Cruise Control to the traffic flow (Bernat Bosch, 2017): investigates the influence of the Cooperative Adaptive Cruise Control (CACC) to the traffic flow.

2. INTRODUCTION

Every day is more frequently to hear about autonomous vehicles. Companies as *Google* or *Tesla* have already built models, doing that they become a reality. So, sooner than expected they will start being able to be find in the conventional highway. But what mathematical algorithms are the ones capable of replacing, or even improving, the human driving? The thing is, that if it was a robot the one in charge of taking all the driving decisions, they would be more accurate. Even if they were capable of sharing information between them, their decision-making data would be extremely precise.

This project consists on creating *Matlab-Simulink* algorithms that implement mathematical equations and try to simulate autonomous driving model. This model includes the design of a track and several autonomous vehicles circulating on it. Based on previous researches that contemplated scenarios as: circulation of only autonomous vehicles with some heterogenic characteristics, circulation of autonomous and humans' vehicles, or sharing-information vehicles, it was decided to, taking those as an initial point, it will be designed a mathematical model that could be used as a tool to make them more realistic.

So, knowing that, there were defined the objectives that would have to have the program originated in this project to include diversity and accuracy to autonomous vehicles simulations.

2.1. PROJECT OBJECTIVES

To define the project objectives, it was searched the principal lacks in the projects exposed in *Section 1.4*. So, the project objectives have been based in what has been considerate that will provide autonomous vehicles simulations of more heterogeneity and realism. Those are:

1. Design a basic model and control of autonomous vehicles following a path in a multi-lane circuit.
2. Improve that basic model in a way that it becomes more realistic, designing a program that gives you the needed data in order to create driving interaction simulations between vehicles.
3. Using the data to Implement some basic interaction between vehicles as ACC or lane changes.
4. Searching an appropriate way to design tracks for the circulation on autonomous vehicles, so the model will be able to be used in scenarios of complex tracks.
5. Create a program that calculates easily the electricity power that is used in simulated scenarios to lay out a tool that can be implemented in the future.

3. MODELLING AND CONTROL OF VEHICLES FOLLOWING A PATH

To simulate an autonomous vehicle following a path it has been created different programs based on the report (1) referenced.

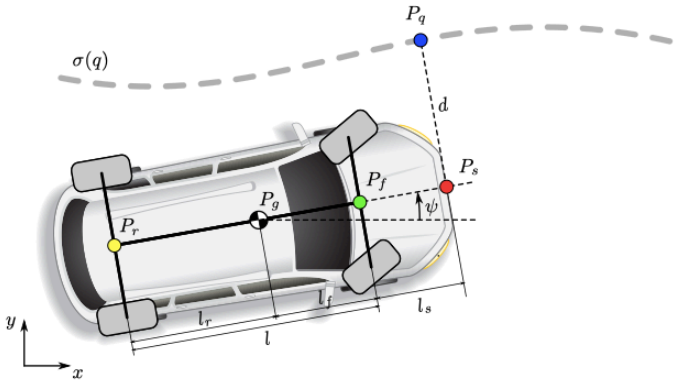


Figure 3.1. View of vehicle variables that are used in the modelling and control of vehicles following a path *(1)*.

Those programs designed calculate different data needed for the control, and follow the next structure:

- Vehicle dynamics: includes an algorithm that implements the basic equations to simulate a vehicle in motion.
- Lane Tracking: includes the algorithms responsible of making that a vehicle follows a defined lane. This part can be subdivided in two:
    - Minimum perpendicular distance: includes an algorithms that, given a location data of a vehicle and a path calculates the minimum perpendicular distance between them.
    - Steering wheel: include the control that allows the vehicle to move in the precise angle direction to follow the path.

3.1. VEHICLE DYNAMICS

The procedure to make possible the simulation of vehicles moving forward has been separated in two parts:

1. The implementation of dynamics equations in a *Matlab* function.
2. The output data control in *Simulink* of the function.

3.1.1. DYNAMICS EQUATIONS

The dynamics equations can be define using the *Figure 3.2* (1):

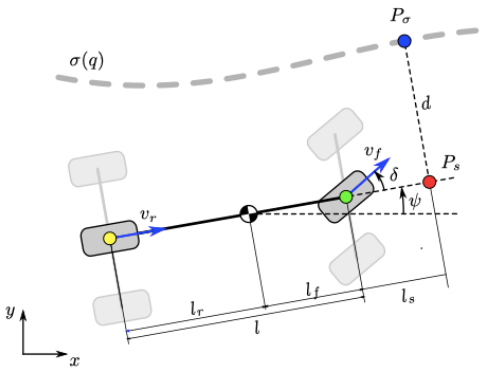


Figure 3.2. View of the vehicle variables that were used in the vehicle dynamics calculations *(1)*.

The dynamics equations are the following (1):

$$\dot{x_f} = v_f \cos(\psi + \delta)$$

$$\dot{y_f} = v_f \sin(\psi + \delta) \qquad \textbf{(Eq. 3.1)}$$

$$\dot{\psi} = \frac{v_f}{l} \cdot \sin(\delta)$$

Where:

x_f , y_f	Point where the vehicle is located (middle frontal wheels point). (m)
v_f	Vehicle speed (m/s)
ψ	Vehicle direction angle respect the horizontal axis (rad).
δ	Angle between the frontal wheels' direction of a vehicle and psi (rad). It would be the direction that will take the steering wheel.
l	Vehicle's length (m).

These equations are implemented in a function named *fcn*, (*Appendix B.1*) and its variables are:

VARIABLE	INPUT / OUTPUT	DESCRIPTION	UNITS
l	Input	Vector formed by the vehicle's lengths.	m
vf	Input	Vector that contains the vehicle's velocities.	m/s
delta	Input	Vector that contains the vehicle's frontal wheel angle directions.	rad
psi	Input	Vector that contains the angle direction of the vehicle. (ψ in figure 4.1).	rad
dxf	Output	Vector formed with the x term of vehicle's vector velocities.	m/s
dyf	Output	Vector formed with the y term of vehicle's vector velocities.	m/s
dpsi	Output	Vector formed with psi vehicle angles.	rad

Table 3.1. Variables of the *Matlab* function *fcn*.

3.1.2. OUTPUT DATA CONTROL

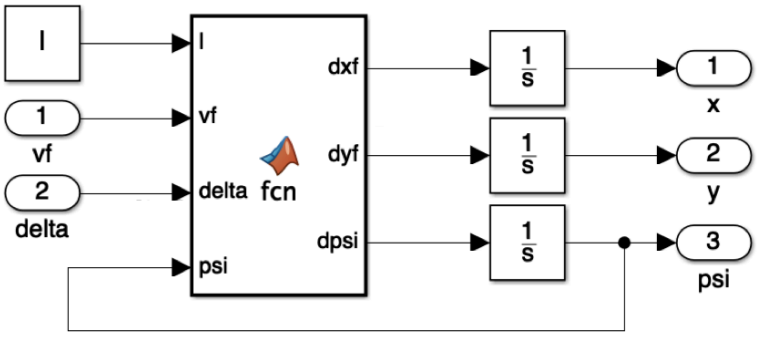


Figure 3.3. Vehicle dynamics block control in *Simulink.*

The output data of the Matlab function *fcn* is integrated, using initial conditions, to get new vehicles position (x, y) and direction (psi). About the input data, l is a constant, delta regulation is explained in *Section 3.2.2* and vf can be a constant (as it is understood in this section) or a variable. In that case, its control is explained in *Section 4.1*.

3.2. LANE TRACKING

The objective of the lane tracking is that the functions related to this part can be implemented in multi-lanes tracks. So, the model created to allowing autonomous vehicles in motion to follow a specific lane, given a multi-lane track consists in:

1 Given an initial position conditions, calculate the minimum perpendicular distance between vehicles and their specific lanes.
2 Creating a steering wheel PI controller, that given different distances allow to know the right steering wheel angle for each of them, to achieve vehicles circulation in several lanes.

3.2.1. MINIMUM PERPENDICULAR DISTANCE BETWEEN A VEHICLE AND A LANE

To calculate the minimum perpendicular distance between circulating vehicles and lanes allows to get the input variable for steering wheel controller.

The equation that defines the calculation of the minimum perpendicular distance between a vehicle (from the midpoint of the front of the vehicle, where it is expected that a sensor would be placed) to one lane point, is the following (1):

$$d = -\sin(\psi) \cdot \left(\sigma_x(q^*) - x_f\right) + \cos(\psi) \cdot \left(\sigma_y(q^*) - y_f\right) \quad \textbf{(Eq. 3.2)}$$

Where:

$\boldsymbol{\sigma_x(q^*)}$, $\boldsymbol{\sigma_y(q^*)}$	x and y lane's point respected to which is calculated the distance. (m)
$\boldsymbol{q^*}$	Parameter of the function that defines the lane. Using it in the lane's function it is obtained the point $\sigma_x(q^*)$, $\sigma_y(q^*)$, the perpendicular and closest lane's point respect the middle frontal part of the vehicle.

To find the q* value it is used the next equation (1):

$$\cos(\psi) \cdot \left(\sigma_x(q^*) - x_f\right) + \sin(\psi) \cdot \left(\sigma_y(q^*) - y_f\right) - ls = 0 \quad \textbf{(Eq. 3.3)}$$

Where:

$\boldsymbol{ls}$	Distance from the point P_f ($\boldsymbol{x_f}$, $\boldsymbol{y_f}$) to P_s ($\boldsymbol{x_s}$, $\boldsymbol{y_s}$) (m). *Figure 3.2.*

How it can be seen in the *Equation 3.3* is important that the functions that define the lanes are parameterized and, as the program used to simulate the functions is *Matlab*, it will be easier if they are expressed as polynomials. That is because *Matlab* can easily give a numerical solution to polyonomies. So, the control of the vehicles following lanes is based in tracks formed by polynomials. **That is the reason why is fundamental that the track design** (*Section 5*) **is carried out with polynomials.**

So, if it is assumed that the function that defines the lanes is a grade's n polynomial, the *Equation 3.3* can be developed:

$$\begin{aligned}\cos(\psi) \cdot \left(C_{n_x} \cdot q^{*n} + C_{n-1_x} \cdot q^{*n-1} + \cdots + C_{1_x} \cdot q^* + C_{0_x} - x_f\right) + \sin(\psi) \\ \cdot \left(C_{n_y} \cdot q^{*n} + C_{n-1_y} \cdot q^{*n-1} + \cdots + C_{1_y} \cdot q^* + C_{0_y} - y_f\right) - ls = 0\end{aligned} \quad \textbf{(Eq. 3.4)}$$

$$\begin{aligned}\left(\cos(\psi) \cdot C_{n_x} + \sin(\psi) \cdot C_{n_y}\right) q^{*n} + \left(\cos(\psi) \cdot C_{n-1_x} + \sin(\psi) \cdot C_{n-1_y}\right) * q^{*n-1} \\ + \cdots + \left(\cos(\psi) \cdot C_{1_x} + \sin(\psi) \cdot C_{1_y}\right) * q^* \\ - \left(\cos(\psi) \cdot C_{0_x} + \sin(\psi) \cdot C_{0_y} + ls\right) = 0\end{aligned} \quad \textbf{(Eq. 3.5)}$$

Where:

$\boldsymbol{C_i}$	Coefficient i of the polynomial that defines a lane.

This polynomial, how it has been said, can be easily numerically solved in *Matlab*, and the solution will indicate all the q^* values that fulfill *Equation 3.3*. Once it is had the solutions, it is followed the next procedure:

1 It is selected the real q^* between 0 and 1.
2 It is calculated the distance (*Eq. 3.2*) for the q^*choosen in the previous point.
3 The q^* solution is the one that gives the minimum distance.

The program designed to implement the equations and processes explained in this section is named *DistanceTrackVehicle,* it can be found in *Appendix B.1*, and the variables used are the following:

VARIABLE	INPUT / OUTPUT	DESCRIPTION	UNITS
C	Input	Matrix formed by all the lanes' functions coefficients.	m
xf	Input	Vector that contains the coordinate x of the vehicles position.	m
yf	Input	Vector that contains the coordinate y of the vehicles position.	m
psi	Input	Vector that contains the angle direction of the vehicles. (ψ in figure 4.1).	rad
DMIN	Output	Minimum perpendicular distance between each vehicle and a lane.	m

Table 3.2. Variables of the *Matlab* function *DistanceTrackVehicle*.

There is a consideration that has to be made about this function:

- Notice, that in the Matlab implementation (*Figure 3.4*), the roots (q^*) that will be "valid" will be the ones which imaginary part is less or equal to 0.001. That is because when this program was tested for the first times (with the "valid" roots having an imaginary part equal to 0), it was noticed that for polynomials formed by periodical coefficients, there could be solutions where all the roots where imaginary. In all that cases, there was one solution with an absolute imaginary part value around 10^{-3}, so it was decided to give an error margin when selecting the real roots.

```
%Formation of QESTREAL (matrix formed by ones and a considerable big
%number (as 10^5)).
%If a number between 0 and 1 of the QEST matrix is real, the QESTREAL
%matrix will have a 1 in the same position of the analyzed number.
%Otherwise, it will have a 10^5.
QESTREAL=zeros(size(QEST,1),size(QEST,2));
for S=1:size(QEST,2)
    for N=1:size(QEST,1)
        if imag(QEST(N,S))<=0.001 & QEST(N,S)<=1 & QEST(N,S)>=0
            qestreal=1;
        else
            qestreal=10^5;
        end
        QESTREAL(N,S)=qestreal;
    end
end
```

Figure 3.4.Part of the *Matlab* function *DistanceTrackVehicle* that shows how the q^*valid values are the ones between 0 and 1 and the one with an imaginary part smaller than 0,001.

3.2.2. STEERING WHEEL CONTROLLER FOR MULTILANE TRACKS

This PI controller returns the steering wheel direction angle delta (δ in *Figure 3.2*), given the minimum perpendicular distance between a vehicle and its lane. It regulates δ to make plausible that the vehicle approximates, to the extent possible, to the lane's point located in the minimum perpendicular distance

To make possible the multi-lane characteristic that was wanted to give to this program it is added an error to the minimum distance reference. If the track had only a single lane, the controller objective would be returning a δ that reduce the minimum perpendicular distance to zero. With the multi-lane property, output data δ is a vector (each δ value, for each vehicle), and its objective distance reduction depends on the vehicle's lane (the intention is not reducing the distance to 0, but to reducing to 0 plus an error). That error (that is a vector) is calculated according to vehicle's lane and width's lane. Given a vehicle i, its error is:

$$error_i = lane_i \cdot width \qquad \textbf{(Eq. 3.6)}$$

Where:

$\boldsymbol{error_i}$	Distance reduction error for a vehicle. (m)
$\boldsymbol{lane_i}$	Vehicle's lane number. $lane_i = 0,1, \ldots$
$\boldsymbol{width}$	Lane's width. (m)

The controllers for implementing them in a single-lane track and a multi-lane track are:

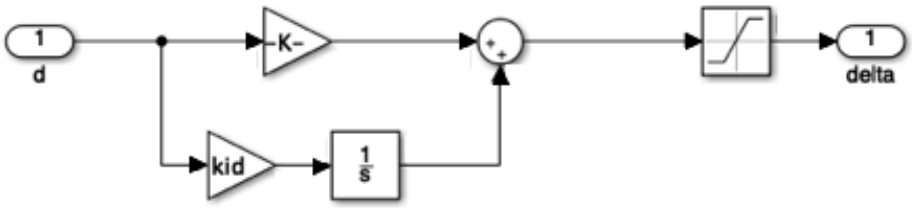


Figure 3.5. PI controllers of single lane model

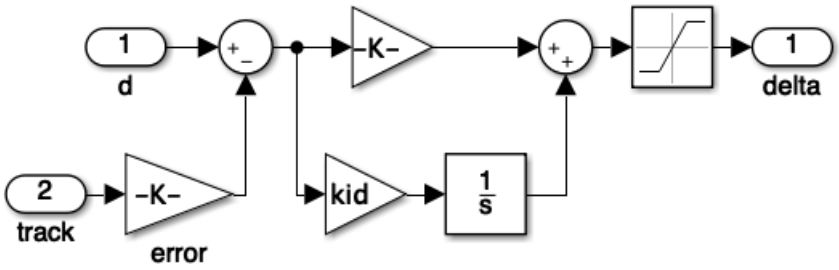


Figure 3.6. PI controllers of multi-lane mode.

The inclusion of a saturator in the controller responds to the necessity of the realistic fact that there is a maximum vehicle steering wheel angle. The port 2 (*Figure 3.6*), label as track, refers to the vector formed by the lanes number in the *Equation 3.6*. The lane's width is directly implemented in the *Simulink* configuration of the error. Lane's vector can be, as the velocity, a constant (which is considered in this section) or a variable. In that case it is explained in *Section 4.2.*

The final PI controller variables are:

Proportional gain (Kp)	0,1
Integral gain (Ki)	10^{-3}
Saturation	0,083π

Table 3.3. PI controller variables.

The gains in the controller have been defined following the trial and error method, and the saturations value was estimated in 15°.

The final block model in *Simulink* that simulates vehicles following a multi-lane track is the next one:

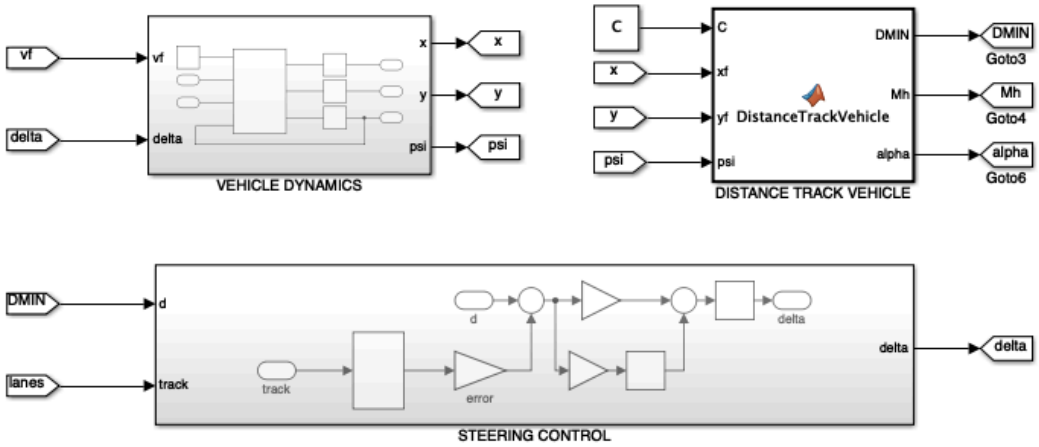


Figure 3.7. Block model in *Simulink* that simulates vehicles following a multi-lane track.

The Mh and alpha outputs of the *DistanceTrackVehicle* function will be explained in *Section 4.1.* And the bloc that can be seen in the Steering Control PI controller *(Figure 3.7)* between the lane input (label as track) and the gain block error will be commented in *Section 4.3*.

4. MODELATION AND CONTROL OF VEHICLES CIRCULATION INTERACTIONS

The primordial objective of this project is to make more realistic the previous models that had been evaluated for this type of autonomous vehicles modelling. That means recreate circulation interactions between vehicles. To do that it is necessary finding good way to set out a principal program that collects all the data needed to perform the interactions. This is the principal achievement in this project. Then, this data has been tasted in specific situations that have been studied in a basic way, but the importance is in the calculation of the driving interactive data.

The more basic interactions that has decided to implement, using the obtention driving interactive data program, in this project have been:

- Speed Control: its principal objective is controlling the vehicles velocity using the ACC system. Includes a mathematical algorithm that recreates the cruising speed function to prevent possible accidents between vehicles.
- Lane Changing: as its name indicates, its function is controlling the vehicles lane changing so they can circulate with their maximum speed. It respects the European according of overtake for the left lane.

4.1. DRIVING INTERACTIVE DATA OBTAINTION

The way that has been lay out this data obtention has been to set up that in every instant it was necessary to know all the distances between vehicle and vehicle and the difference between their angle direction. This creates two data matrixes that would give all the necessary information to create interaction driving simulations.

The equations that calculates this data are the following:

Given a car A and a car B, the distance (Mh) and the angle direction difference (alpha) between them, are:

$$Mh_{AB} = \sqrt{\left(x_{f_A} - x_{f_B}\right)^2 + \left(y_{f_A} - y_{f_B}\right)^2} \quad \textbf{(Eq. 4.1)}$$

$$alpha_{AB} = \tan^{-1}\left(\frac{y_{f_B} - y_{f_A}}{x_{f_B} - x_{f_A}}\right) - \psi \quad \textbf{(Eq. 4.2)}$$

That is calculated in *Matlab* in the *DistanceTrackVehicle (Appendix B.1)* function, and the information is stored in Mh and alpha matrix respectively.

Mh	Output	Matrix that contain the distance between a vehicle and the others in the track.	m
alpha	Output	Matrix that contain the angle direction difference between a vehicle and the others in the track.	rad

Table 4.1. Output variables (Mh and alpha) obtained in the *Matlab* function *DistanceTrackVehicle*.

With this data it is possible to calculate all the other ones that are specific necessary for particular interactions driving simulations, as the followings.

4.2. SPEED CONTROL

The speed control that has been implemented responds to the following equations (2):

To define the speed $v_{d_\in}$ for each vehicle, it has been used the following formula:

$$v_{d_\in}(h) = \begin{cases} v_{max} + \dfrac{h - h_c}{T} & h - h_c \leq -\varepsilon \\ v_{max} - \dfrac{1}{4 \cdot \in \cdot T} \cdot (h - h_c - \varepsilon)^2 & |h - h_c| < \varepsilon \\ v_{max} & h - h_c \geq \varepsilon \end{cases} \quad \textbf{(Eq. 4.3)}$$

Where:

$\boldsymbol{v_{max}}$	Maximum vehicle speed (m/s)
$\boldsymbol{h}$	Bumper-to-bumper distance between two cars (m). Given a car L (frontal car), witch bumper position is x_L and its length is L, and a vehicle F (backward car) which position is x_F, the bumper to bumper distance h between them would be:

	$h = x_L - x_F - L$
$\boldsymbol{h_c}$	Safety distance between a car and its frontal vehicle (m). $h_C = h_o + T * v_{max}$ Where h_O is the vehicle desired net bumper-to-bumper minimum distance to the precedent vehicle.
$\boldsymbol{T}$	Safety time headway for a vehicle when following others. (s)
$\boldsymbol{\varepsilon}$	$\varepsilon = 0{,}1$. It is a parameter evaluated in report (2) that makes that the vehicles speed does not suddenly reduce but, it is reduced gradually.

Notice that h will be found using the Mh and alpha matrixes. Being i the vehicle that is being evaluated to found its h, the method to obtain h_i it will be:

1. Selecting the vehicles in the matrix that are in the same lane.
2. Selecting the vehicles which alpha's cosine are positive. (that means that they will be in front of the vehicle).
3. Selecting the vehicle which distance (stored in Mh matrix) is minimum.

The *Matlab* function that responds to *Eq. 4.1* and the procedure to obtain the h_i for each vehicle is named *SpeedControl (Appendix B.2):*

VARIABLE	INPUT / OUTPUT	DESCRIPTION	UNITS
Mh	Input	Matrix formed by the distance between all the vehicles and all the other ones.	m
alpha	Input	Matrix formed by the angle direction difference between all the vehicles and all the other ones.	rad

Vmax	Input	Vector that contains the vehicle's maximum speed.	m/s
ho	Input	Vector that contains the desired bumper-to-bumper minimum distance of each vehicle.	m
T	Input	Vector formed with the safety time headway of each vehicle.	S
lanesret	Input	Vector that contains the number's lane where circulates each vehicle simulated.	-
vf	Input/Output	Vector that contains the real vehicle's velocity.	m/s
vfd	Output	Vector that contains the vehicle's velocity wanted.	m/s
vfront	Output	Vector that contains the real velocity of the vehicle in front of another.	m/s
posfront	Output	Vector that contains the position (in all the data vectors) of the vehicle in front of another	-

Table 4.2. Variables of the *Matlab* function *SpeedControl*.

Notice that this function gives information about all the vehicles in front of others. This data is also obtained with the Mh and alpha matrixes. Its obtention procedure is really simple using *Matlab*, so it can be found in *Appendix B.2.* On the other hand, the difference between the real speed (vf) and the

wanted speed (vfd) will be explained in *Section 4.2.1.* About the input data lanesret in this section will be considerate as the constant lanes, but in *Section 4.3* it will be developed.

4.2.1. WANTED VELOCITY AND REAL VELOCITY

It has been decided, to make more realistic the speed control model, that the change of velocity will not be instantaneous. So, to implement that it has been created a simple bloc of control:

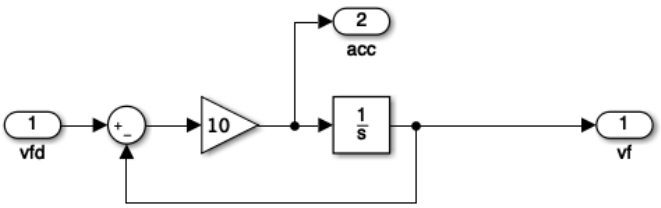


Figure 4.1. Bloc of control to modify the velocity in a way that its change is not instantaneous.

So, the wanted velocity is the resulting in the *Speed Control* function and, the real velocity, is the one resulted for applying to it, the delay caused by the bloc of control. The gain (*Figure 4.1*) has been obtained using the trial and error method and the output of the port 2 is the acceleration of the vehicles, data implemented in *Section 4.3.*

The final blocs of control in *Simulink* related to ACC are the following ones:

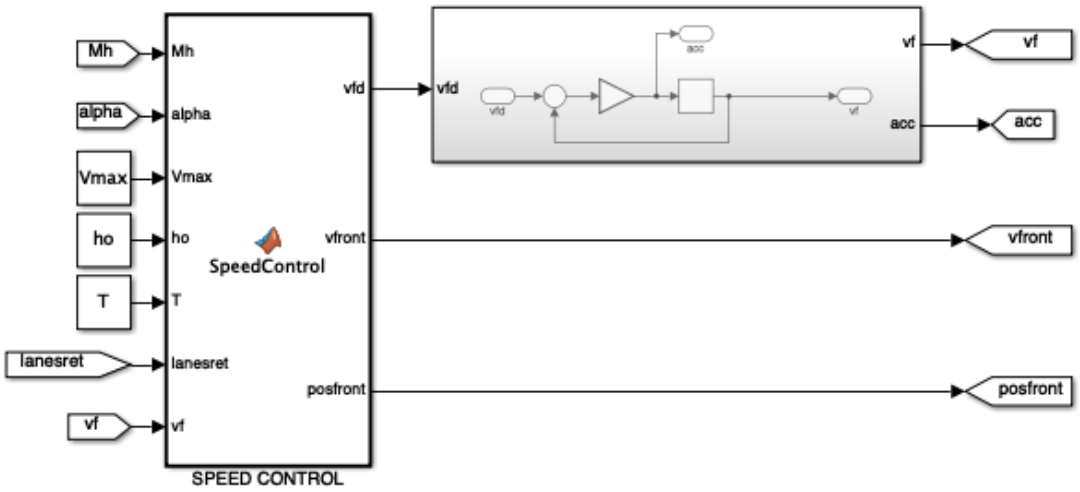


Figure 4.2. Speed Control blocs in Simulink.

4.3. LANE CHANGING

The lane changing program that has been created is very basic. It responds to simple conditions that request some data. This data has been once again founded with Mh and alpha matrixes. The information specifically needed for lane changing, for each vehicle involved in a simulation, is:

- Distance with the closest front vehicle of its right lane (hRFront).
- Distance with the closest back vehicle of its right lane (hRBack).
- Distance with the closest front vehicle of its left lane (hLFront).
- Distance with the closest back vehicle of its left lane (hLBack).
- Vehicle acceleration.

The vehicle's acceleration is found in *Figure 4.1* and the other data will be obtained, for each vehicle, following the next procedure:

1. Search the vehicles in the right lane of the vehicle's lane.
2. Search the vehicles in the left lane of the vehicle's lane.
3. Search the vehicles which cosine is positive (that means that they will be in front of the vehicle).
4. Search the vehicles which cosine is negative (that means that they will be back of the vehicle).
5. Select the vehicles, using the previous points, according the data that wants to be calculated.
6. Select the minimum distance in Mh of the vehicles selected in the previous point.

To implement this procedure, it has been created a *Matlab* program named *DistanceData (Appendix B.2)*, which variables are:

VARIABLE	INPUT / OUTPUT	DESCRIPTION	UNITS
Mh	Input	Matrix formed by the distance between all the vehicles and all the other ones.	m
alpha	Input	Matrix formed by the angle direction difference between all the vehicles and all the other ones.	rad

hLFront	Output	Vector formed by the distances between each vehicle and its closest front vehicle in its left lane.	m
hLBack	Output	Vector formed by the distances between each vehicle and its closest back vehicle in its left lane.	m
hRFront	Output	Vector formed by the distances between each vehicle and its closest front vehicle in its right lane.	m
hRBack	Output	Vector formed by the distances between each vehicle and its closest back vehicle in its right lane.	m

Table 4.3. Variables of the Matlab function *DistanceData*

With this data, has been imposed some conditions to the vehicles for changing its lane:

$$lane_i = \begin{cases} lanesret_i + 1 \ \ if \begin{cases} vf_i < Vmax_i \ \wedge \\ hLBack_i > hLBackmin \ \wedge \\ acc_i \leq 0 \ \wedge \\ hLFront > hLFrontmin \end{cases} \\ lanesret_i - 1 \ if \begin{cases} vf_i < Vmax_i \ \wedge \\ hRBack_i > hRBackmin \ \wedge \\ hRFront > hRFrontmin \end{cases} \\ lanesret_i \end{cases} \quad \textbf{(Eq. 4.4)}$$

Where:

$lane_i$	Vehicle's new lane.
$lanesret_i$	Vehicle's previous lane.
acc_i	Acceleration of the vehicle (m/s^2)
$hLBackmin$	Minimum distance that has to have a vehicle with its back-left lane vehicle to proceed with the left lane changing. (m)
$hLFrontmin$	Minimum distance that has to have a vehicle with its front-left lane vehicle to proceed with the left lane changing. (m)
$hRBackmin$	Minimum distance that has to have a vehicle with its back-right lane vehicle to proceed with the right lane changing. (m)
$hRFrontmin$	Minimum distance that has to have a vehicle with its front-right lane vehicle to proceed with the right lane changing. (m)

The *Equation 4.4* has been implemented in a Matlab program named *LaneChange* (*Appendix B.2*), which variables have been:

VARIABLE	INPUT / OUTPUT	DESCRIPTION	UNITS
acc	Input	Vector formed by the acceleration of each vehicle.	m/s^2
vf	Input	Vector that contains the real vehicle's velocity.	m/s
lanesret	Input	Vector that contains the previous lane of the vehicles.	-

hLFront	Input	Vector formed by the distances between each vehicle and its closest front vehicle in its left lane.	m
hLBack	Input	Vector formed by the distances between each vehicle and its closest back vehicle in its left lane.	m
hRFront	Input	Vector formed by the distances between each vehicle and its closest front vehicle in its right lane.	m
hRBack	Input	Vector formed by the distances between each vehicle and its closest back vehicle in its right lane.	m
Vmax	Input	Vector that contains the vehicle's maximum speed.	m/s
lane	Output	Vector that contains the number's lane where circulates each vehicle simulated.	-

Table 4.4. *Matlab* function *LaneChange* variables.

To get the lanesret, it has been used a memory block in *Simulink*:

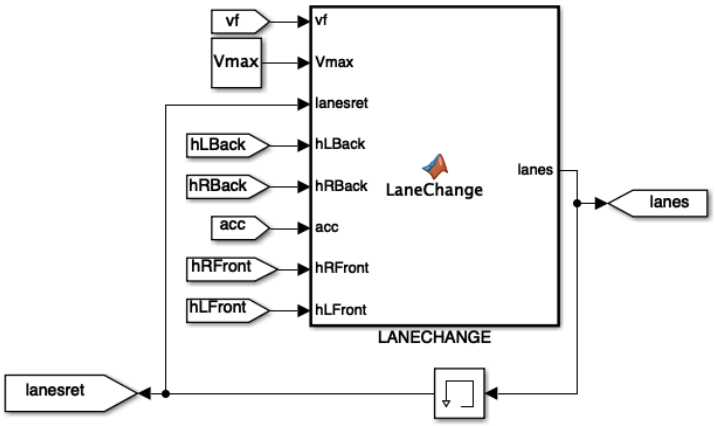


Figure 4.3. *LaneChange* block in *Simulink.*

5. TRACK DESIGN

The track design has to be proposed in a way that was defined by polynomial functions, as it was a requisite lay out in the *Section 3.2.* Another objective was to build the program so it could be used in complex tracks, understanding as complex tracks those build using more than one function. Those are the reasons why it was decided to design it using Bézier Curve's.

To supply this last point, the program to design the track had to have the possibility of uniting Bézier Curves so the union was differentially continuous. And finally, it was needed an algorithm that represented graphically the track, with its various lanes.

5.1. BÉZIER CURVES

An intuitive way to describe Bézier Curves would be as functions that, expressed in polynomial form, show coefficients build from a given initial control points. The advantage that have is that, graphically, it can be seen how the curve path "follows" the polynomial formed by those points, named control points. Even though, they will only contain the first and the last ones that have been initially given.

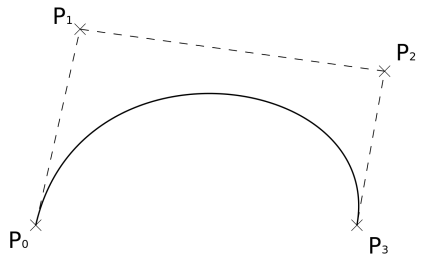


Figure 5.1. Bézier Curve formed by the control points P^0, P^1, P^2 and P^3. **(3)**

Bézier Curves are given by the following equation:

$$\boldsymbol{B(t) = \sum_{i=0}^{n} \binom{n}{i} \cdot (1-t)^{n-i} * t^{i} * P_i} \qquad \textbf{(Eq. 5.1)}$$

Where:

n	Constant which represents the number of control points that are initially given to the function. It is said that the curve will be n degree. N ∈ ℵ

i	Number of referred control point. i=0,1..., n-1, n. i ∈ ℵ
t	Parameter whose function depend on. t ∈ [0,1]
P_i	Control point i (x_i, y_i). $P_i=P_0$, P_1..., P_{n-1}, P_n. (m)
$\binom{n}{i}$	Binomial coefficient, whose equation is $\frac{n!}{i!*(n-i)!}$
$B_i^n(t) = \sum_{i=0}^{n} \binom{n}{i} \cdot (1-t)^{n-i} \cdot t^i$	Bernstein polynomial, that allows to express Bézier Curves in a simple way. $B(t) = B_0^n(t) \cdot P_0 + B_1^n(t) \cdot P_1 + \cdots + B_n^n(t) \cdot P_n$

5.1.1. BÉZIER CURVES EXPRESSED AS POLYNOMIALS

During the vehicles modelling it was seen that in Matlab the easiest way to work with functions if those that are expressed with polynomial form, so given any value, Matlab can evaluate easily the function.

Bézier Curves can be expressed in polynomial form, as follows:

$$\boldsymbol{B(t) = C_0 \cdot t^n + C_1 \cdot t^{n-1} + \cdots + C_{n-1} \cdot t + C_n} \quad \textbf{(Eq. 5.2)}$$

Whre the coefficients can be calculated with:

$$C_i = \left(\sum_{f=1}^{n-i+1} \frac{-1^{f-1+n-i}}{(f-1)! \cdot (n-i-f+1)!} \cdot P_f \right) \cdot \left(\sum_{m=1}^{n-i} n - m \right) \quad \textbf{(Eq. 5.3)}$$

Where:

$\boldsymbol{P_f}$	Vector formed by the coordinates xf, yf, so the solution of the equation is a vector too. (m)
$\boldsymbol{C_i}$	Vector formed by C_{xi} i C_{yi}.

The program that has been implemented is named *Bcoefficients* (appendix B.3) and has the following input and output data:

VARIABLE	INPUT / OUTPUT	DESCRIPTION	UNITS
P	Input	Matrix formed by all the control points that belong to the Béziers curves' coefficients that want to be calculated.	m
n	Output	Number of the control points that has a curve.	-
C_X, C_Y	Output	Matrix with the coefficients of all the input Bézier Curves.	m

Table 5.1. *Bcoefficients* Matlab function variables.

There is a consideration that has to be made about it:

- By giving a matrix as a result and as an input data, it is impossible to calculate curves' coefficients with different control points number.

5.1.2. DIFFERENTIAL CONTINUITY APPROXIMATION IN BÉZIER CURVES

When the track has been designed an essential requisite has been that the diverse Bézier Curves unions that conformed it has to be differentially continuous. So, it has been proposing the following conditions to make that this happen:

Given two Bézier Curves named $B_1(t)$ and $B_2(t)$ and formed by the control points matrix M_1 and M_2,

$$M_1 = \begin{pmatrix} x_0^1 & x_1^1 & \dots & x_n^1 \\ y_0^1 & y_1^1 & \dots & y_n^1 \end{pmatrix} \qquad M_2 = \begin{pmatrix} x_0^2 & x_1^2 & \dots & x_n^2 \\ y_0^2 & y_1^2 & \dots & y_n^2 \end{pmatrix}$$

They will be approximately differently continuous if:

$$x_0^2 = x_n^1 \,, \;\; y_0^2 = y_n^1 \qquad \textbf{(Eq. 5.4)}$$

$$y_1^2 = \frac{y_n^1 - y_{n-1}^1}{x_n^1 - x_{n-1}^1} \cdot x_1^2 + y_{n-1}^1 - \frac{y_n^1 - y_{n-1}^1}{x_n^1 - x_{n-1}^1} \cdot x_{n-1}^1 \qquad \textbf{(Eq. 5.5)}$$

In words, for two Bézier curves to be approximately continuous:

1. The first second curve's control point (P_0^2) must be the same as the last first curve's control point (P_n^1). (Eq. 5.4)
2. The second second curve's control point (P_1^2) must be aligned with the last (P_n^1) and penultimate first curve's control point (P_{n-1}^1). (Eq. 5.5)

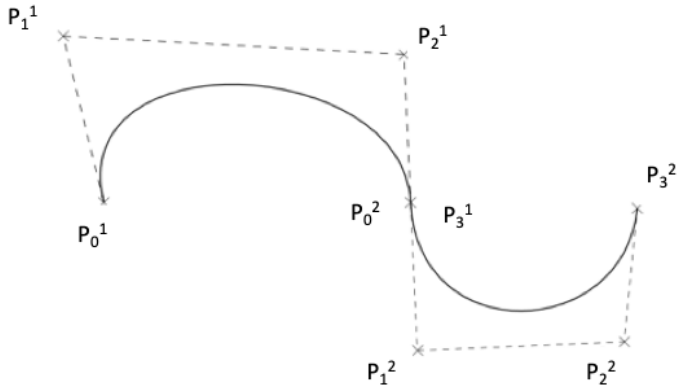


Figure 5.2.Curve formed by two approximately differently continuous Bézier Curves *(4)*.

5.1.3. SPECIFIC CASE: TWO CONTINUOUS BÉZIER CURVES FORMING A CLOSED PATH

The finality of doing an elaborated circuit does not respond to the objective of designing the more complex track but, preparing the program to be implemented in future more elaborated pathways. That was the reason why it was decided to create an algorithm that recreates the simple complex type of tracks: two continuous Bézier Curves that form a closed path.

To do that there is a program named *BezierContinues (Appendix B.3)* that calculates the control points of a second curve given one, so the resulted figure have the appearance of a more or less circular track. The procedure to plan the program is located in the appendant A. The variables of this program are:

VARIABLE	INPUT / OUTPUT	DESCRIPTION	UNITS
V1	Input and Output	Matrix formed by the given curve's control points.	m
V2	Output	Matrix formed by the calculated curve's control points.	m

Table 5.2. *BezierContinues* Matlab function variables.

5.1.4. BEZIER CURVE'S GRAPHIC REPRESENTATION

To represent graphically Bézier Curve it is only needed a function that gives t values to equation 3.1. To plot Bézier Curves in Matlab, it is used a function named *Bcurve (Appendix B.3)*, whose variable are:

VARIABLE	INPUT / OUTPUT	DESCRIPTION	UNITS
C	Input	Matrix formed by all the curve's control points.	M
npts	Input	Number of points that want to be represented.	-
t	Output	Values that are t in Eq. 3.2 to represent the function.	-
b	Output	Matrix formed by all the solutions of Eq. 3.2 given the t values.	M

Table 5.3. *Bcurve* Matlab function variables.

But this representation only allows to plot the curves that are defined by its function. So, to graphic parallel curves that represent the multi-lanes of the track, it would be needed its functions. As those are mathematically very difficult to get, it has been opted for using another method, explained in the following sections.

5.2. GRAPHIC REPRESENTATION: PARALLEL CURVES

To represent more than one lane in a graphic, it is necessary to create parallel curves using as base a Bézier Curve. The method that has been used only allows to know specific points that will conform the parallel curves, but it does not give their functions. To get those it would be necessary to follow another method, which mathematically is highly difficult to resolve and, as in this project the parallel curves are only needed to be plot in a graphic (as the vehicles circulate for different lanes due to the control, as it is explained in *Section 3.2.2*), it has been used a simpler way to represent them.

This method consists in using specific points of a curve, to get its parallels:

Given a matrix $\alpha = \begin{pmatrix} x_0 & x_1 & \dots & x_t \\ y_0 & y_1 & \dots & y_t \end{pmatrix}$, that graphically represents a curve, the method that has to be followed to obtain another matrix $\beta = \begin{pmatrix} \beta x_0 & \beta x_1 & \dots & \beta x_t \\ \beta y_0 & \beta y_1 & \dots & \beta y_t \end{pmatrix}$, parallel to α is the next one:

1 Being i the column index, calculate the matrix $\alpha' = \begin{pmatrix} x'_0 & x'_1 & \dots & x'_t \\ y'_0 & y'_1 & \dots & y'_t \end{pmatrix}$:

$$\alpha'_1 = \frac{\alpha_2 - \alpha_1}{2}$$

$$\alpha'_t = \frac{\alpha_t - \alpha_{t-1}}{2} \qquad \textbf{(Eq. 5.6)}$$

$$\alpha'_{2\dots t-1} = \frac{\alpha_{t+1} - \alpha_{t-1}}{2}$$

2 Normalize the vector (x'_i, y'_i) and compose the matrix $\alpha'_n = \begin{pmatrix} {x_n}'_0 & {x_n}'_1 & \dots & {x_n}'_t \\ {y_n}'_0 & {y_n}'_1 & \dots & {y_n}'_t \end{pmatrix}$:

$$x_{n_i}' = \frac{x'_i}{|\alpha_n'|} = \frac{x'_i}{\sqrt{{x'_i}^2 + {y'_i}^2}}$$

(Eq. 5.7)

$$y_{n_i}' = \frac{y'_i}{|\alpha_n'|} = \frac{y'_i}{\sqrt{{x'_i}^2 + {y'_i}^2}}$$

3 Rotate the vector $({x_n}'_i, {y_n}'_i)$, $\pi/2$ radians and form the matrix $n = \begin{pmatrix} nx_0 & nx_1 & \dots & nx_t \\ ny_0 & ny_1 & \dots & ny_t \end{pmatrix}$:

$$(nx_i, ny_i) = (-{y_n}'_i, {x_n}'_i) \qquad \textbf{(Eq. 5.8)}$$

4 Multiply n matrix for the parallel distance d and add the initial position α to obtain β:

$$(\beta x_i, \beta y_i) = (d \cdot nx_i + x_i, d \cdot ny_i + y_i) \quad \textbf{(Eq. 5.9)}$$

This method has been implemented in a Matlab function named *Parallel (Appendix B3)*, which variables are:

VARIABLE	INPUT / OUTPUT	DESCRIPTION	UNITS
b	Input	Matrix formed by all the curves points.	m
numberoflanes	Input	Vector that contains the number of lanes.	-
amplada	Input	Lane width.	m
alphatot	Output	Matrix formed by all the parallel curves points.	m

Table 5.4. Parallel Matlab function variables.

6. POWER AND ENERGY CONSUMTION OF THE VEHICLES' FLEET

The power consume calculation has been focused in creating a program that, given any simulated scenario, compute the power electricity consumed by the vehicles and gives energy information.

It has been considerate that the vehicles were electric (as it is foreseeable that the future autonomous cars will be electric), in concrete it has been set the *Nissan Leaf* as the prototype model. The characteristics of this vehicle in a specific road and the properties of itself are:

Mass (m)	1521 kg
Road Grade (θ)	0 rad
Rolling resistance parameters (Cr, c1 and c2)	Cr=1,75 c1=0,0328 c2=4,575
Vehicle's frontal area (Af)	2,3316 m^2
Vehicle's aerodynamic drag coefficient (CD)	0,28
Driveline efficiency [1]($\eta_{driveline}$)	92%
Electric motor efficiency ($\eta_{electricmotor}$)	91%

Table 6.1. Characteristics of the vehicle (*Nissan Leaf*) and road prototype used. *(5)*

The equation used to calculate the power consumed by the wheels (P_{wheels}) for a vehicle is the following (5):

$$P_{wheels}(t) = \left(ma + mg \cdot \frac{\cos(\theta)\, Cr}{1000}(c1 \cdot v(t) + c2) + \frac{1}{2}\rho_{air} \cdot A_f \cdot C_D \cdot v(t)^2 + mg \cdot \sin(\theta) \right) v(t) \quad \textbf{(Eq. 6.1)}$$

[1] Efficiency related to the loses sum of components which are delivering the engine power to the wheels.

Where:

P_{wheels}	Wheels power consumed (W)
m	Mass (kg)
a	Acceleration (m/s^2). If the vehicle is decelerating can acquire negative values.
g	Gravity (m/s^2). It has been supposed that $\boldsymbol{g} = 9{,}8066$
θ	Road grade.
C_r	Rolling resistance parameters.
c1	
c2	
v	Vehicle Speed (m/s).
ρ_{air}	Air density (kg/m^3). It has been supposed that $\boldsymbol{\rho_{air}} = 1{,}2256$
A_f	Vehicle's frontal area (m^2).
C_D	Vehicle's aerodynamic drag coefficient.

The total power consumed by the motor (PEM), if the vehicle is not deaccelerating is (5):

$$PEM(t) = \frac{P_{wheels}(t)}{\eta_{driveline} \cdot \eta_{electricmotor}} \quad \textbf{(Eq. 6.2)}$$

Where:

PEM	Motor power consumed (W)
$\eta_{driveline}$	Driveline efficiency.
$\eta_{electricmotor}$	Electric motor efficiency.

If the vehicle is deaccelerating, its function mode is the regenerative braking mode[1]. In that case the motor power consumed will be define by (5):

$$PEM(t) = \frac{-P_{wheels}(t)}{\eta_{driveline} \cdot \eta_{electricmotor}} * \eta_{rb}(t) \quad \textbf{(Eq. 6.3)}$$

Where:

η_{rd}	Regenerative braking efficiency.

The regenerative braking efficiency is given by the following equation (5):

$$\eta_{rb}(t) = \left(e^{\frac{0.0411}{|a(t)|}}\right)^{-1} \quad \textbf{(Eq. 6.4)}$$

Where:

a	Deacceleration (m/s^2)

So, the energy usage (W) by a vehicle and the average energy usage ($\bar{W}$) in Joules would be calculated as:

$$W = \int_0^t PEM(t) * dt$$

$$\bar{W} = \frac{\sum_{i=1}^{nvehiclles} W_i}{nvehicles} \quad \textbf{(Eq. 6.5)}$$

The program that has been implemented in *Matlab* is named *EnergyData (Appendix B.4)*:

VARIABLE	INPUT / OUTPUT	DESCRIPTION	UNITS
acc	Input	Vector that contains the vehicle's acceleration.	m/s^2
vf	Input	Vector that contains the real vehicle's velocity.	m/s

[1] When the vehicle is deaccelerating energy flows from the wheels to the motor. In this case, the power at the electric motor is lower than the power at the wheels and the power is assumed to be negative.

PEM	Output	Power consumed by the motor for a vehicle.	J

Table 6.2. *EnergyData* variables.

In *Simulink* it has been used an integrator block to obtain the energy usage of each vehicle:

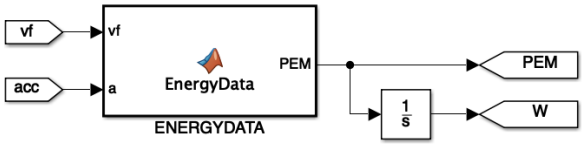


Figure 6.1. *EnergyData* block in *Simulink.*

7. SIMULATIONS RESULTS

To prove that the algorithms and programs work given a specific scenario it has been created a simulation given this initial data:

TRACK DESIGN	
V1 (m) (*Table 5.2*)	$\begin{pmatrix} 0 & 0 \\ 100 & 100 \\ 200 & 100 \\ 300 & 300 \\ 400 & 0 \end{pmatrix}$
numberoflanes (*Table 5.4*)	2
width (m) (*Eq. 3.6*)	10
VEHICLES CHARACTERISTICS (5 vehicles)	
Vmax (m/s) (*Table 4.4)*	$(130 \quad 50 \quad 140 \quad 40 \quad 60)$
V0 (m/s) (Initial speed)	$(0 \quad 0 \quad 0 \quad 0 \quad 0)$
Lanes0 (Initial position lanes)	$(0 \quad 0 \quad 0 \quad 0 \quad 0)$
T (s) (*Table 4.2*)	$(0{,}21 \quad 0{,}20 \quad 0{,}12 \quad 0{,}65 \quad 1{,}56)$
ho (m) (*Table 4.2*)	$(1{,}34 \quad 2{,}48 \quad 3{,}97 \quad 2{,}59 \quad 2{,}16)$
l (m) (*Table 3.1)*	2
delta0 (rad) (initial delta)	$(0 \quad 0 \quad 0 \quad 0 \quad 0)$
x0 (m) (initial x position)	$(76{,}77 \quad 60{,}61 \quad 44{,}44 \quad 28{,}28 \quad 12{,}12)$
y0 (m) (initial y position)	$(61{,}79 \quad 50{,}48 \quad 38{,}53 \quad 25{,}68 \quad 11{,}60)$
psi0 (rad) (initial psi position)	$(0{,}52 \quad 0{,}52 \quad 0{,}52 \quad 0{,}52 \quad 0{,}52)$
MODEL CHARACTERISTICS	
Simulation time (s)	50

Max Step Size	10^{-2}

Table 7.1 - Simulated scenario initial data.

It has to be said that these values, does not pretend to be realistic but give a good appreciation of the different results. The next sections will analyse some results, but the total of them are in *Appendix C*

7.1. VEHICLE DYNAMICS AND LANE TRACKING RESULTS

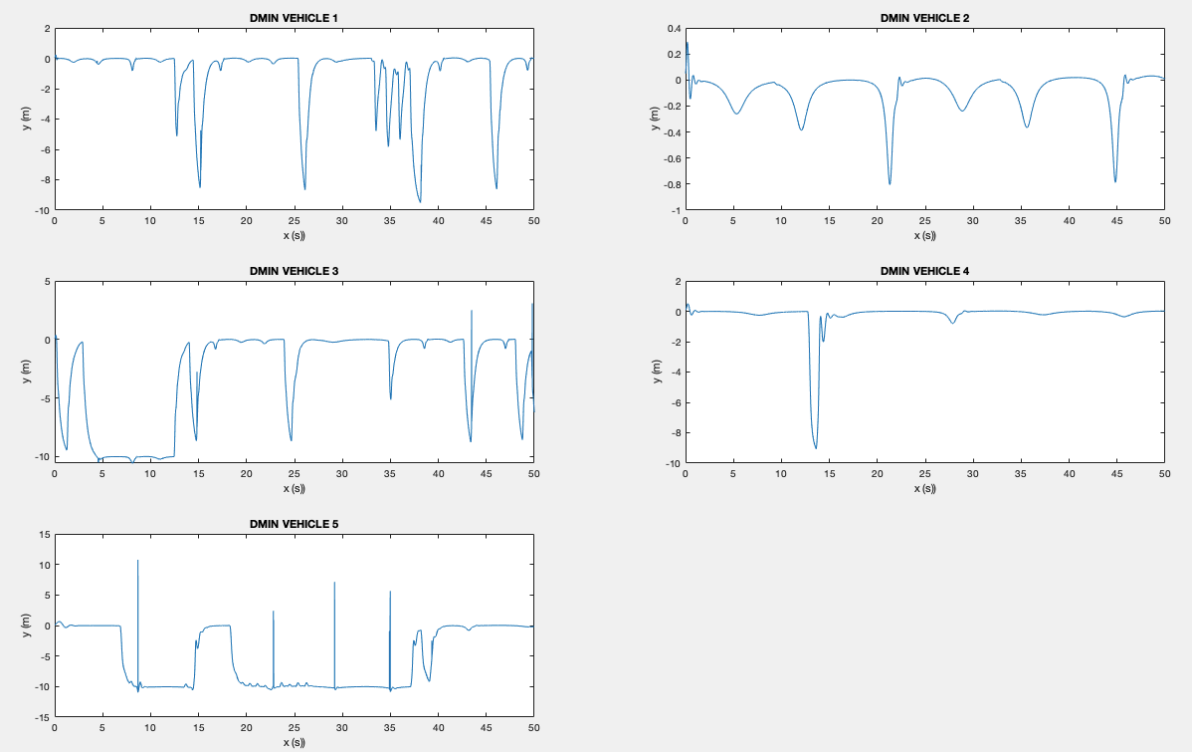


Figure 7.1- Vehicle Dynamics and Lane Tracking results.

As it can be seen, the vehicles 1, 3, 4 and 5 will change its lane in some moment as the minimum distance reference change between 0 and -10[1]. If it is analysed all the vehicles individually it can be taken the next conclusions:

- **Vehicle 1:** This is one of the fasters vehicles, and with graphic can show how it overtakes other vehicles the seconds 15, 26, 37 and 46. The second 13 and the instants between 33 and 37 it

[1] The right lane is the 0 referenced lane, and the left lane, which is used by the vehicles to overtake other vehicles, is the -10 referenced lane.

can be seen how it tries to change its lane, but it stays far from being complete in the other lane. This indicates that the lane input is changing really fast because the lane changing conditions are oscillating in the limit for allowing the change.

- **Vehicle 2:** This vehicle is interesting to analysed because it does not change its lane. So, in the graphic can be seen perfectly when minimum distance between the vehicle and the lane is bigger due to control. In those seconds the vehicle surpasses the most curved part of the track. It can be seen perfectly how goes two times around the circuit.
- **Vehicle 3:** This graphic is very similar to the vehicle 1 graphic. It shows how vehicle 3 also change its lane several times to surpass other vehicles. The difference is that during some instances (between 5 and 11 second) it will be stabilized in the lane referenced as -10.
- **Vehicle 4:** This vehicle is the slower one, so it should not change its lane. But in second 13 it can be seen how it does it. This is probably because another vehicle, that has recently changed its lane getting to vehicle's 4 front, has slowed down vehicle's 4 velocity due to the speed control. This shows how the lane change algorithm has to be perfectionated.
- **Vehicle 5:** It could be said that this vehicle is the more decontrolled one. In seconds 8, 23, 29 and 35 its minimum distance acquires values above 0. It is another prove that lane change control fails sometimes.

7.2. SPEED CONTROL RESULTS

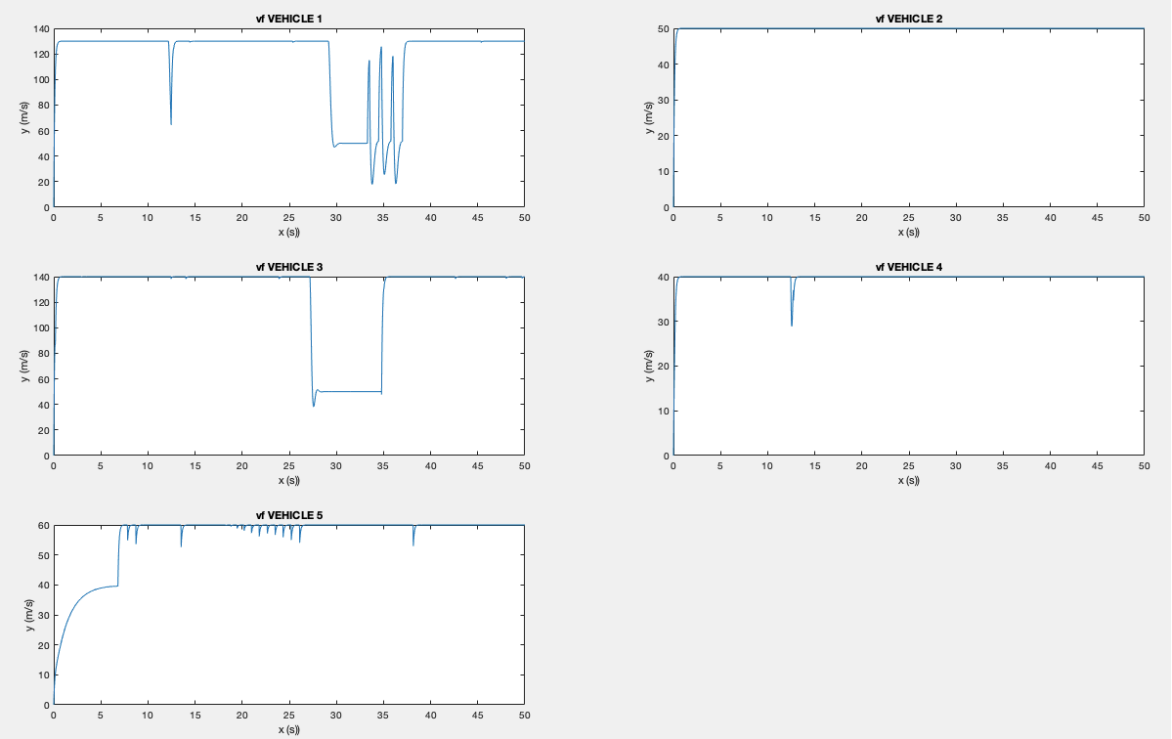


Figure 7.2- Speed Control result

The speed control graphics of the vehicle show perfectly how before a lane changing its speed has went down. Sometimes it can be seen what vehicle is in front of others because the speed of a faster vehicle is stabilized in a velocity value of another. This shows how the speed control works.

- **Vehicle 1:** The vehicle's 1 speed is slowed down by vehicle 5 in second 13, just before it changes its lane and between the instances 29 and 37. It can be seen, how it has been said in the analysis of the previous graphic, how the speed was oscillating and therefore its lane change was doing it too.
- **Vehicle 2:** Vehicle's 2 speed is constant, and it is its maximum. That is the reason why it never changes its lane.
- **Vehicle 3:** it is only slowed down in the interval between 27 and 35 but in the previous graphic it has been seen how it changes its lane more times. That could be because the changes are really fast when it slows down a little bit, and that makes that the speed goes up fast. This is because the lane changes condition (in terms of space with the vehicles that surround the vehicle) to make a change in those instants are the suitable.
- **Vehicle 4:** this graphic confirms the conclusions that has been proposed in the previous analysis.
- **Vehicle 5:** first this vehicle is back of vehicle 4, but then the previous graphic shows how it surpasses it and acquire its maximum speed.

7.3. LANE CHANGING RESULTS

Although the lane change has been explained with the other graphics, this one shows which lane had every vehicle in each instance. 1 is the left lane and 0 is the right one.

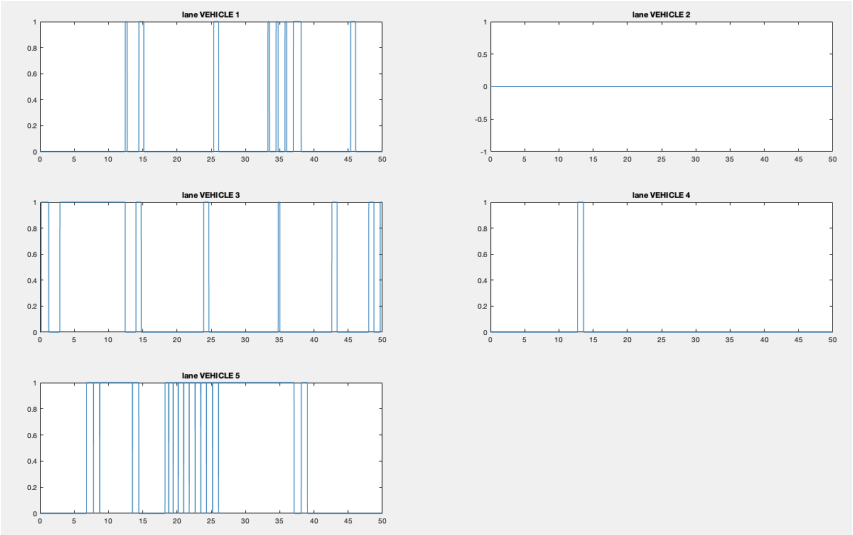


Figure 7.3. Lane Change resu

CONCLUSIONS

To conclude it could be said that this project has result to become a series of tools to implement in future simulated scenarios. They will help to give reality to them; therefore, they will be more precise and will give results that adjust better to real vehicles circulation. The way that has been lay out the interactions data between vehicles, will allow other projects, that study in depth the driving relations between vehicles, to have a solid base to start. This will enrich the investigations about vehicles circulation simulations.

The first tool that has result of creating this project has been a simple program that moves a vehicle following a given path. This path is a parametrized polynomial function, which takes to the second tool: the Bézier Curves. The study that has been made about them will allow to create complex tracks, to be studied in future scenarios. But it could be said that the more important program, has become the one related with obtention of the data about vehicles interactions. This algorithm has allowed to implement speed control simulations and to start creating a base for the lane changing. The last tool that has been designed it has been the energy data tool. Though it is very simple, it will be very useful to traffic and vehicles driving efficiency simulations studies.

It has to be said that all these implements can be improved, the *Simulink* controls can be more adjusted so the result of the simulations can even be more realistic and, especially the lane change algorithm, can be more sophisticated, but they are a good way to progress in the vehicles simulation field.

ECONOMIC ANALYSIS

The economic analysis in this project can be broken down in several parts:

- Equipment: Such as the computer used to program the simulations or the office supplies.
- Transport: The budged used to buy the train tickets for the university-home journeys.
- Energy resources: Energy costs to charge the computer used in this project.
- Labour costs: The salary per hour for a junior engineer.

EQUIPMENT

	PRICE (€)
APPLE MACBOOK AIR MQD32Y/A 13.3”	936
OFFICE SUPPLIES (APPROXIMATION)	10
MATLAB LICENSE	154
TOTAL	**1100**

Table. 1 - Material budget's project. *(6) (7)*

TRANSPORT

	PRICE (€)
T-JOVE 3 ZONES	199,2
TOTAL	**199,2**

Table. 2 - Transport budget's project. *(8)*

ENERGY RESOURCES

To calculate the energy resources, it has been necessary the computer and power greed characteristics:

Power Consumption (Wh)	54
Medium electricity price per day (€/kWh)	0,112 (July)
	0,106 (August)
	0,110 (September)

Table. 3- Computer and power greed characteristics. *(9) (10)*

	€/Kwh	Computer consumption (Wh)	Hours per month	Price (€)
July	0,112	54	240	1,455
August	0,106		240	1,368
September	0,110		240	1,427
			TOTAL	**4,25**

Table. 4 - Energy resources budget's project.

LABOUR COST

Junior engineer estimated salary (€/h)	Time worked (h)	Price (€)
8,5	720	6120
	TOTAL	**6120**

Table. 5 – Labour cost budget's project. *(11)*

TOTAL

	PRICE (€)
EQUIPMENT	1100
TRANSPORT	199,2
ENERGY RESOURCES	4,25
LABOUR COST	6120
TOTAL	**7423,5**

Table. 6 - Total budget's project.

BIBLIOGRAPHY

1. **Dòria-Cerezo, A.** *Simulations of a vehicle following a path using Bézier curves.* 2018.

2. **Arnau Dòria-Cerezo, Josep M. Olm, Ernest Benedito, Domingo Biel and Victor Repecho.** *A first order sliding mode-based adaptive cruise controller.* 2018.

3. **Wikipedia.** [Online] [Cited: 5 9 2019.] https://en.wikipedia.org/wiki/B%C3%A9zier_curve#/media/File:Bezier_curve.svg.

4. **Harvey, Andrew.** Andrew Harvey'blog. [Online] https://andrewharvey4.wordpress.com/2009/12/02/computer-graphics-notes/.

5. **Chiara Fiori, Kyoungho Ahn, Hasham A Rakra.** *Power-based Electric Vehicle Energy Consumption Model: Model Development and Validation .* 2016.

6. **MediaMarkt.** MediaMarkt. [Online] September 2019. https://www.mediamarkt.es/es/product/_apple-macbook-air-mqd32y-a-13-3-intel%C2%AE-core%E2%84%A2-i5-5350u-8-gb-ram-128-gb-ssd-intel%C2%AE-hd-graphics-6000-1368059.html?gclid=Cj0KCQjwz8bsBRC6ARIsAEyNnvpix9qWWExyEH54a0tPjRcI9mVatDaNw8KNa_tyGceSl2QfQDt3TSMaAh.

7. **Mathworks.** Mathworks. [Online] September 2019. https://es.mathworks.com/store/link/products/home/new.

8. **FGC.** fgc. [Online] September 2019. https://www.fgc.cat/xarxa-fgc/bitllets-integrats/#1504260679452-76620344-2d06.

9. **Apple.** Apple. [Online] September 2019. https://support.apple.com/kb/SP753?locale=en_US.

10. **Tarifaluzhora.** tarifaluzhora. [Online] September 2019. https://tarifaluzhora.es/?tarifa=normal&fecha=2019-09-18.

11. **glassdoor.** glassdoor. [Online] September 2019. https://www.glassdoor.es/Sueldos/barcelona-junior-engineer-sueldo-SRCH_IL.0,9_IM1015_KO10,25.htm?countryRedirect=true.

APPENDIX A

This appendant will contain the procedure that was followed to create a program that made two continuous Bézier curves forming a closed figure, given one.

1 It was determined the different scenarios that the first curve could be found in a graphic and would affect the estimation of the second one in a way that, the equations needed to calculate the second curve's control points, would be different:

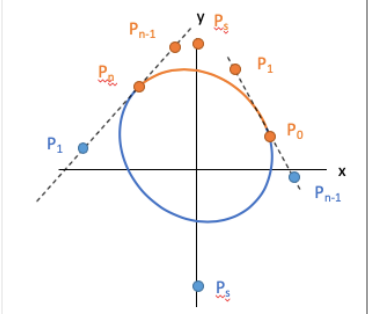
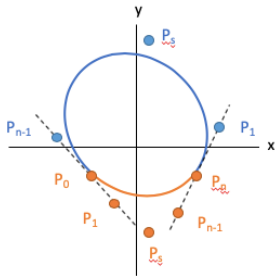
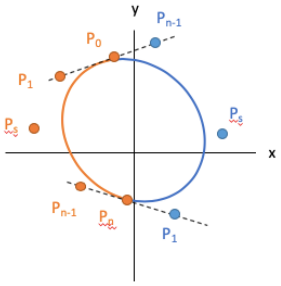
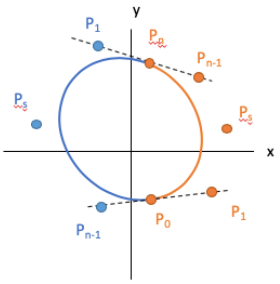
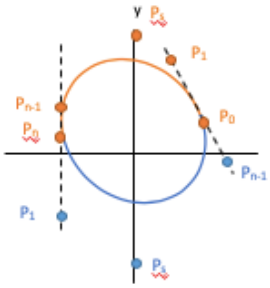
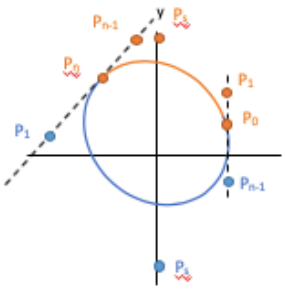
		
Scenario A.I	Scenario A.II	Scenario B.I
		
Scenario B.II	Scenario C	Scenario D

Table 0.1. Scenarios that will affect the equations of the program studied in this chapter. The given curve would be the orange one and the calculated with the program *BezierContinues* the blue one.

2 The equations that describe every scenario were define:

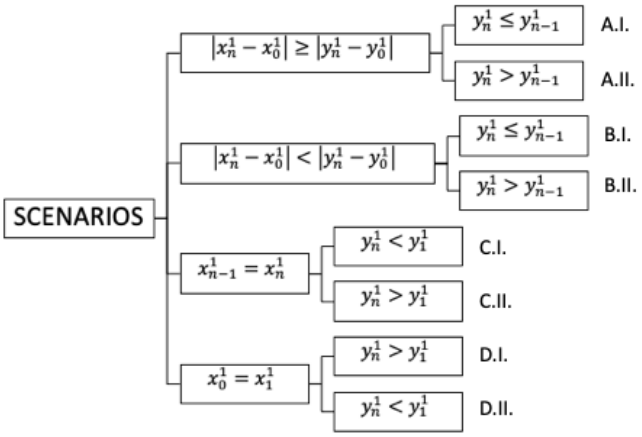


Figure. 1. Equations that defines scenarios of the program *BezierContinues*.

3 Finally, the equations that were used in every scenario to calculate the second curve's control points were established:

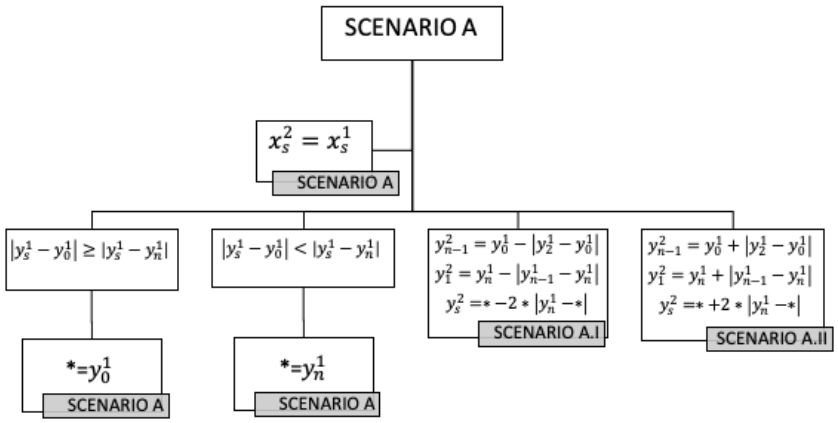


Figure. 2. Equations that were used in scenario A in the program *BezierContinues.*

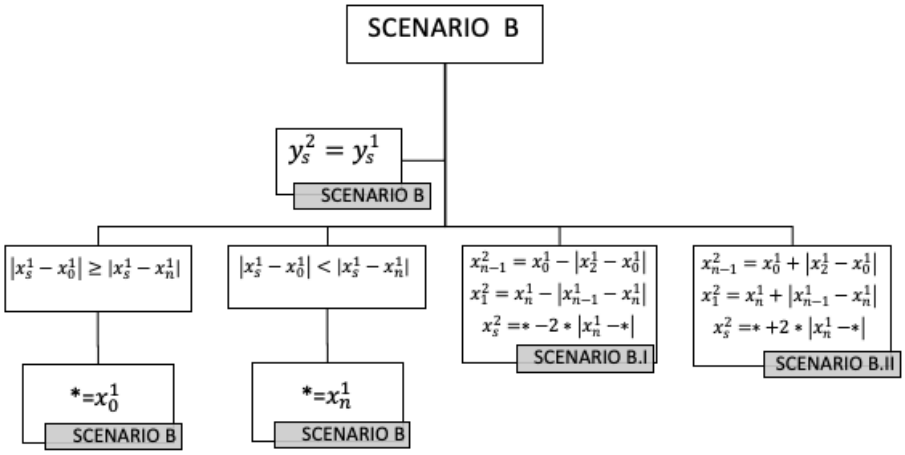


Figure. 3. Equations that were used in scenario B in the program *BezierContinues.*

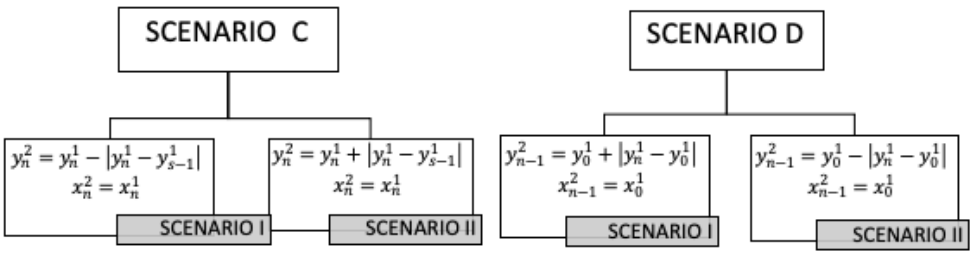


Figure. 4. Equations that were used in scenarios C and D in the program BezierContinues.

APPENDIX B

This appendant will expose all the program codes that has been described in this project.

B1. LANE TRACKING AND VEHICLE DYNAMICS PROGRAM CODES

```
function [DMIN,Mh,alpha]   = DistanceTrackVehicle(C, xf, yf, psi)
%INITIAL DATA DECLARATION
nvehicles=length(xf);
DMIN=zeros(nvehicles,1);
alpha=zeros(nvehicles,nvehicles);
Mh=zeros(nvehicles,nvehicles);

%EQUATIONS TO CALCULATE FOR EACH CAR
for T=1:nvehicles

%Formation of equation 8
COS=cos(psi(T)).*C(1:2:3,:);
SIN=sin(psi(T)).*C(2:2:4,:);
M1=COS+SIN;
M2=-cos(psi(T))*xf(T)-sin(psi(T))*yf(T)-0.5;
s=size(M1,2);
M1(1:2,s)=M2+M1(1:2,s);

%Solution of equation 8 and roots matrix QEST formation (the "empty"
%matrix spaces, due to the different number coefficients that can have
%each curve, will be filled with a negative complex number as -1+2i)
QEST=ones(length(M1),2).*-1+2i;
for N=1:size(M1,1)
    qest=roots(M1(N,:));
    QEST(1:size(qest,1),N)=qest;
end

%Formation of QESTREAL (matrix formed by ones and a considerable big
%number (as 10^5)).
%If a number between 0 and 1 of the QEST matrix is real, the QESTREAL
%matrix will have a 1 in the same position of the analyzed number.
%Otherwise, it will have a 10^5.
```

```
QESTREAL=zeros(size(QEST,1),size(QEST,2));
for S=1:size(QEST,2)
   for N=1:size(QEST,1)
        if imag(QEST(N,S))<=0.001 & QEST(N,S)<=1 & QEST(N,S)>=0
            qestreal=1;
        else
            qestreal=10^5;
        end
        QESTREAL(N,S)=qestreal;
   end
end

%Perpendicular distance between a car and various track points (the ones
%given by the roots of the QEST matrix). Equation 5.
D=zeros(size(QEST,1),size(QEST,2));
x=1;
for S=1:size(QEST,2)
    sigx=polyval(C(x,:),QEST(:,S));
    sigy=polyval(C(x+1,:),QEST(:,S));
    D(:,S)=real(-sin(psi(T)).*(sigx-xf(T))+cos(psi(T)).*(sigy-yf(T)));
    x=x+2;
end

%Distances obtained multiplied by 1 if the point of the circuit used has
%been obtained with a root that fulfills the appropriate conditions and
%for 10^5 if it does not.
dposs=D.*QESTREAL;

%Formation of a matrix that gives 1 or -1, according to the distance
%sign.
signes=zeros(size(QEST,1),size(QEST,2));
for S=1:size(dposs,2)
    for N=1:size(dposs,1)
```

```
        if dposs(N,S)<0
            signe=-1;
        else
            signe=1;
        end
        signes(N,S)=signe;
    end
end

%Minimum perpendicular distance (absolute value), and its position in
%the dposs matrix.
[minabs,fila]=(min(abs(dposs)));
[MINABS,colum]=(min(minabs));

%Multiplication of the minimum perpendicular distance and its sign.
DMINN=MINABS(1,1)*signes(fila(1,colum),colum(1,1));

%Storage (in a vector) of the minimum distance for each car.
DMIN(T)=DMINN;

h_j=sqrt((xf(T)-xf).^2+(yf(T)-yf).^2);
alpha_j=atan2(yf-yf(T),xf-xf(T))-psi(T);
Mh(T,:)=h_j;
alpha(T,:)=alpha_j;
end
```

Figure. 6. *DistanceTrackVehicle* Matlab code.

```
function [dxf,dyf,dpsi] = fcn(l,vf,delta,psi)

dxf=vf.*cos(psi+delta);
dyf=vf.*sin(psi+delta);
dpsi=vf./l.*sin(delta);
```

Figure. 5. *fcn* Matlab code.

B2. INTERACTION DRIVING PROGRAM CODES

```
function [vfd,vfront,posfront]  = SpeedControl(Mh,alpha,Vmax,ho,T,lanesret,vf)
%INITIAL DATA DECLARATION
eps=0.1;
n=length(Mh);
h=zeros(n,1);
hc=zeros(n,1);
vfd=zeros(n,1);
vfront=zeros(n,1);
posfront=zeros(n,1);
B=zeros(n,1);
%Equations to calculate for each car.
for j=1:n
    %Definition of the vehicles's lane and search of the cars belonging to
    %the same lane.
    Z=lanesret(j);
    S=find(lanesret==Z);

    %hFront=distance between a vehicle and the vehicles in the same lane.
    hFront=sign(cos(alpha(j,S))).*Mh(j,S);

    %Search of the vehicles in front of another (in the same lane).
    ind=find(hFront>zeros(1,length(S)));

    if isempty(ind)
       %If there is any vehicle in front of the studied, the first one will go
       %full speed
        vfd(j)=Vmax(j);
        %If there is not a car in front of the one that is being analized,
        %the speed of the frontal car will be 1e5.
        vfront(j)=10^5;
        posfront(j)=0;
    else
        [h(j),posfront(j)]=min(hFront(ind));
        %If there is a car in front of the one that is being analized,
        %the speed of the frontal car will be:
        vfront(j)=vf(posfront(j));
        %And the speed of the analized car will respond to the speed
        %control equations:
        hc(j)=ho(j)+T(j)*Vmax(j);
        vfd(j)=Vmax(j).*(h(j)-hc(j)>=eps)+...
        (Vmax(j)+(h(j)-hc(j))/T(j)).*(h(j)-hc(j)<=-eps)+...
        (Vmax(j)-1/4/eps/T(j).*(h(j)-hc(j)-eps).^2).*(abs(h(j)-hc(j))<eps);
    end
end
```

Figure. 7. *SpeedControl* Matlab code.

```
function [hLFront,hLBack,hRFront,hRBack]  = DistanceData(Mh,alpha,lanesret,num
%INITIAL DATA DECLARATION
ncotxes=length(lanesret);
h=zeros(ncotxes);
dist=zeros(ncotxes);
hLFront=zeros(1,ncotxes);
hRFront=zeros(1,ncotxes);
hLBack=zeros(1,ncotxes);
hRBack=zeros(1,ncotxes);
i=zeros(1,ncotxes);
for j=1:ncotxes
    %Distance between vehicle and vehicle
    dist(:,:)=sign(cos(alpha(:,:))).*Mh(:,:);

    %Lane of the vehicle
    i(j)=lanesret(j);

    %Positions of the vehicles in the left, in the right, in the front and
    %in the back
    L=find(lanesret==i(j)+1);
    R=find(lanesret==i(j)-1);
    Front=find(dist(j,:)>0);
    Back=find(dist(j,:)<0);

    %datadeclarations
    LFront=zeros(1,min(length(L),length(Front)));
    LBack=zeros(1,min(length(L),length(Back)));
    RFront=zeros(1,min(length(R),length(Front)));
    RBack=zeros(1,min(length(R),length(Back)));

    %Numebers of the closer vehicles in the frontal left lane  and back
    %left lane
    m=1;
```

```
    s=1;
    for a=1:length(L)
        for b=1:length(Front)
            if L(a)==Front(b)
                LFront(1,m)=Front(b);
                m=m+1;
            end
        end
        for b=1:length(Back)
            if L(a)==Back(b)
                LBack(1,s)=Back(b);
                s=s+1;
            end
        end
    end
    LFront=LFront(1,find(LFront(1,:)~=0));
    LBack=LBack(1,find(LBack(1,:)~=0));

    %Numebers of the closer vehicles in the frontal right lane  and back
    %right lane
    m=1;
    s=1;
    for a=1:length(R)
        for b=1:length(Front)
            if R(a)==Front(b)
                RFront(1,m)=Front(b);
                m=m+1;
            end
        end
        for b=1:length(Back)
            if R(a)==Back(b)
                RBack(1,s)=Back(b);
            end
        end
    end
    RFront=RFront(1,find(RFront(1,:)~=0));
    RBack=RBack(1,find(RBack(1,:)~=0));

    %If the vehicle is in the 0 lane.
    if i(j)==0
        hRFront(j)=1e5;
        hRBack(j)=1e5;
        if isempty(LFront)
            hLFront(j)=1e5;
        else
            hLFront(j)=min(abs(dist(j,LFront)));
        end
        if isempty(LBack)
            hLBack(j)=1e5;
        else
            hLBack(j)=min(abs(dist(j,LBack)));
        end

     %If the vehicle is in the last left lane
    elseif i(j)==numberoflanes
        hLFront(j)=1e5;
        hLBack(j)=1e5;
        if isempty(RFront)
            hRFront(j)=1e5;
        else
            hRFront(j)=min(abs(dist(j,RFront)));
        end
        if isempty(RBack)
            hRBack(j)=1e5;
        else
```

```
            hRBack(j)=min(abs(dist(j,RBack)));
        end

    %If the vehicle is in middle lanes
    else
        if isempty(LFront)
            hLFront(j)=1e5;
        else
            hLFront(j)=min(abs(dist(j,LFront)));
        end
        if isempty(LBack)
            hLBack(j)=1e5;
        else
            hLBack(j)=min(abs(dist(j,LBack)));
        end
        if isempty(RFront)
            hRFront(j)=1e5;
        else
            hRFront(j)=min(abs(dist(j,RFront)));
        end
        if isempty(RBack)
            hRBack(j)=1e5;
        else
            hRBack(j)=min(abs(dist(j,RBack)));
        end
    end

end
```

Figure. 8. *DistanceData Matlab* program code.

```
function lanes    = LaneChange(vf,Vmax,lanesret,hLBack,hRBack,acc,hRFront,hLFront)
nvehicle=length(vf);
lanes=zeros(1,nvehicle);
for j=1:nvehicle
    if vf(j)<Vmax(j)-0.1 & hLBack(j)>30 & acc(j)<=0 & hLFront(j)>30
        if lanesret(j)~=1
           lanes(j)=lanesret(j)+1;
        end
    elseif vf(j)>Vmax(j)-0.001 & hRBack(j)>30 & hRFront(j)>30
        if lanesret(j)~=0
           lanes(j)=lanesret(j)-1;
        end
    else
        lanes(j)=lanesret(j);
    end
end
```

Figure. 9. *LaneChange Matlab* program code

B3. TRACK DESING PROGRAM CODES

```
function [n,C] = Bcoefficients(P)
for l=1:2:size(P,2)-1
    V=P(:,l:l+1)
    n=length(V);fn=factorial(n);
    for cnt1=1:n
        aux1=[0;0];
        j=n-cnt1;
        for cnt2=1:j+1
            k=cnt2-1;
            aux1=aux1+(-1)^(k+j)/(factorial(k)*factorial(j-k))*V(cnt2,:)'
        end
        aux2=1
        for m=1:j
            aux2=aux2*(n-m)
        end
        c(:,cnt1)=aux2*aux1
    end
    C(l:l+1,:)=c
end
```

Figure. 10. *Bcoefficients* Matlab code.

```
function [V1,V2]=BezierContinues(V1)
s=length(V1)
V2(1,:)=V1(s,:)
V2(s,:)=V1(1,:)
if abs(V1(s,1)-V1(1,1))>=abs(V1(s,2)-V1(1,2))
    cnt=2
    inco=1
else
    cnt=1
    inco=2
end
if V1(s,cnt)<=V1(s-1,cnt)
    sig=-1
else
    sig=1
end
for n=3:s-2
    if abs(V1(n,cnt)-V1(1,cnt))>=abs(V1(n,cnt)-V1(s,cnt))
        est(n)=V1(1,cnt)
    else
        est(n)=V1(s,cnt)
    end
    V2(n,cnt)=est(n)+sig*abs(V1(s-n+1,cnt)-est(n))
end
V2(3:s-2,inco)=V1(s-2:-1:3,inco)

if V1(s-1,1)==V1(s,1)
    if V1(s,2)<V1(1,2)
        sige=-1
    else
        sige=1
    end
```

```
    V2(2,2)=V1(s,2)+sige*abs(V1(s,2)-V1(s-1,2))
    V2(2,1)=V1(s,1)
end
if V1(1,1)==V1(2,1)
    if V1(s,2)<V1(1,2)
       sige=1
    else
       sige=-1
    end
    V2(s-1,2)=V1(1,2)+sige*abs(V1(1,2)-V1(2,2))
    V2(s-1,1)=V1(1,1)
end
if V1(s-1,1)~=V1(s,1)
    Y1(1,:)=V1(s-1:s,2);
    X1(1,:)=V1(s-1:s,1);
    X1(2,:)=ones(1,2);
    Rtg1=Y1*X1^(-1);
    V2(2,cnt)=V1(s,cnt)+sig*abs(V1(s-1,cnt)-V1(s,cnt))
    if cnt==2
       V2(2,inco)=(V2(2,cnt)-Rtg1(1,2))/Rtg1(1,1)
    else
       V2(2,inco)=V2(2,cnt)*Rtg1(1,1)+Rtg1(1,2)
    end
end
if V1(1,1)~=V1(2,1)
    Y2(1,:)=V1(1:2,2);
    X2(1,:)=V1(1:2,1);
    X2(2,:)=ones(1,2);
    Rtg2=Y2*X2^(-1);
    V2(s-1,cnt)=V1(1,cnt)+sig*abs(V1(2,cnt)-V1(1,cnt))
    if cnt==2
       V2(s-1,inco)=(V2(s-1,cnt)-Rtg2(1,2))/Rtg2(1,1)
       V2(s-1,inco)=V2(s-1,cnt)*Rtg2(1,1)+Rtg2(1,2)
    end
end
end
```

Figure. 11. *BezierContinues* Matlab code.

```
function [t,b] = Bcurve(C,npts)
for l=1:2:size(C,1)-1
    c=C(l:l+1,:)
    t = linspace(0,1,npts);
    n=length(c);
    B=zeros(2,npts);        % Bezier curve initial
    for cnt3=1:n
       j=n-cnt3;
       B=B+t.^j.*c(:,cnt3);
    end
b(l:l+1,:)=B
end
end
```

Figure. 12. *Bcurve* Maltab code.

```
function [alphatot]=Parallel(b,carrils,amplada)
a=1;
for x=1:2:size(b,1)
    B=b([x x+1],:);
    for t=2:length(b)-1
        alpha([x x+1],t)=(B(:,t+1)-B(:,t-1))/2;
    end
    alpha([x x+1],1)=(B(:,2)-B(:,1))/2;
    alpha([x x+1],length(b))=(B(:,length(b))-B(:,length(b)-1))/2;
    for t=1:length(b)
        unit(a,t)=(alpha(x,t)^2+alpha(x+1,t)^2)^(1/2);
        alphanorm([x x+1],t)=alpha([x x+1],t)/unit(a,t);
        alphagir(x,t)=-(alphanorm(x+1,t));
        alphagir(x+1,t)=(alphanorm(x,t));
    end
    a=a+1;
end
r=1
for l=1:max(carrils)
    alphatot(r:size(alphagir,1)*l,:)=alphagir.*(-amplada*l)+b;
    r=r+4
end
```

Figure. 13. *Parallel* Matlab code.

B4. POWER PROGRAM CODES

```
function PEM   = EnergyData(vf,acc)
ncotxes=length(vf);
rendReg=zeros(1,ncotxes);
Pwheels=zeros(1,ncotxes);
PEM=zeros(1,ncotxes);
m=1521;
g=9.8066;
RoadGrade=0;
Cr=1.75;
C1=0.0328;
C2=4.575;
AirDensity=1.2256;
Af=2.3316;
Cd=0.28;
for j=1:ncotxes
    if acc(j)<0
        rendReg(j)=(exp(0.0411/abs(acc(j))))^-1;
    else
        rendReg(j)=0;
    end
    Pwheels(j)=(m*acc(j)+m*g*cos(RoadGrade)*Cr/1000*(C1*vf(j)+C2)..
        +1/2*AirDensity*Af*Cd*vf(j)^2+m*g*sin(RoadGrade))*vf(j);
    PEM(j)=Pwheels(j)/(0.92*0.91)*(acc(j)>=0)...
        -Pwheels(j)/(0.92*0.91)*rendReg(j)*(acc(j)<0);
end
```

Figure. 14. EnergyData Matlab program code.

APPENDIX C

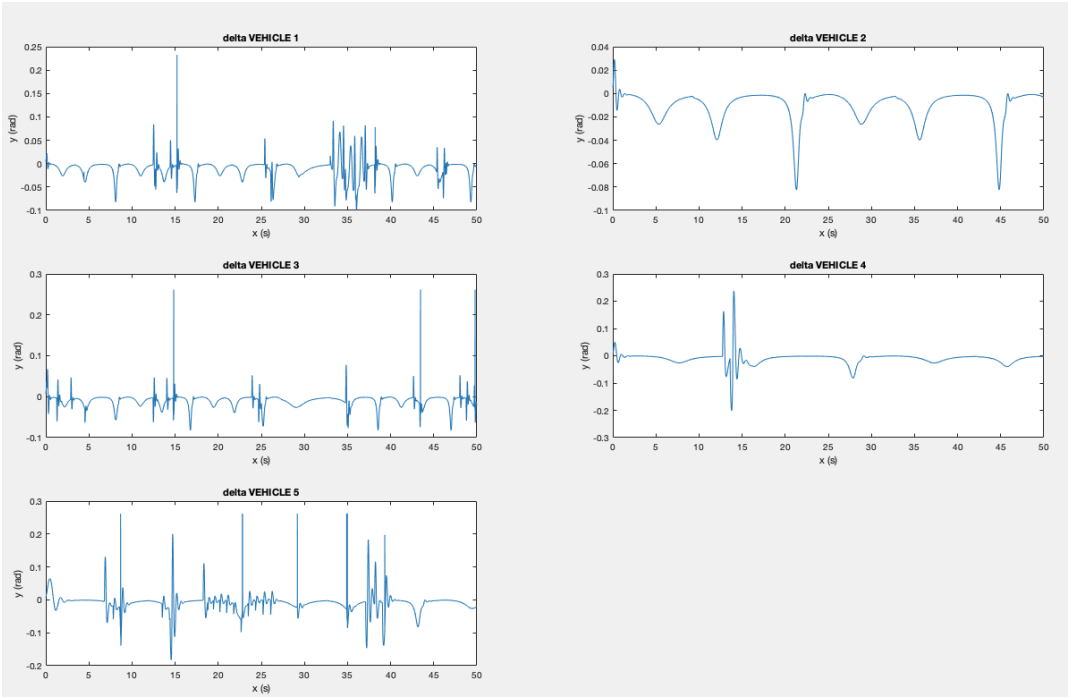


Figure. 15. delta simulation results.

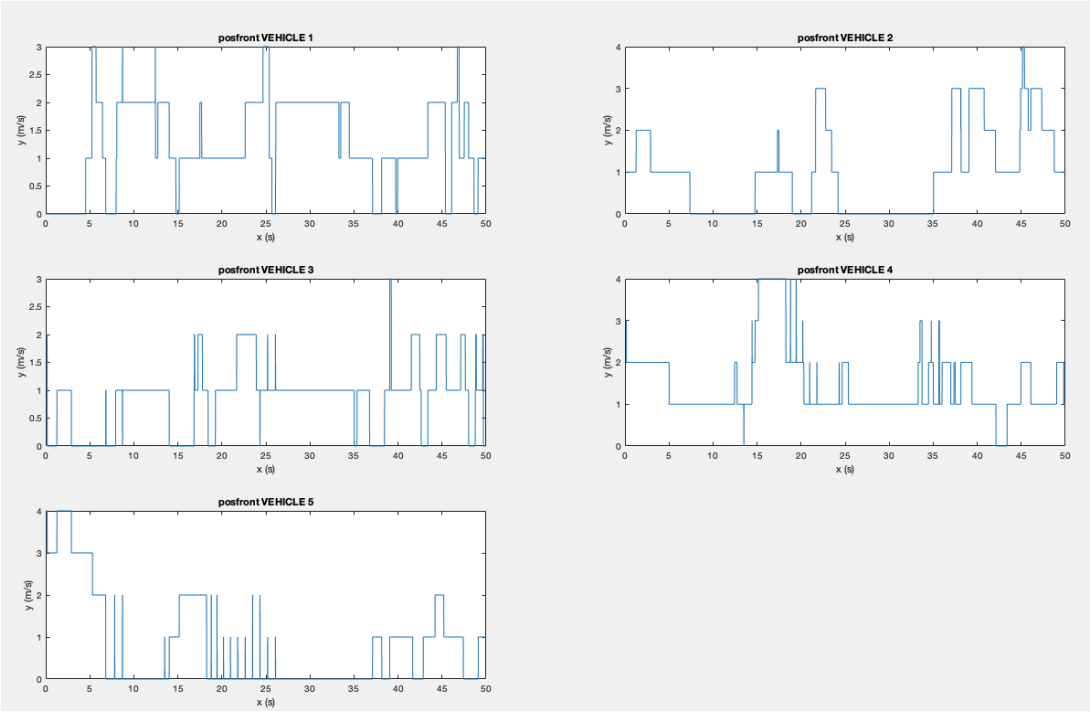


Figure. 16. Vehicle frontal position simulation result.

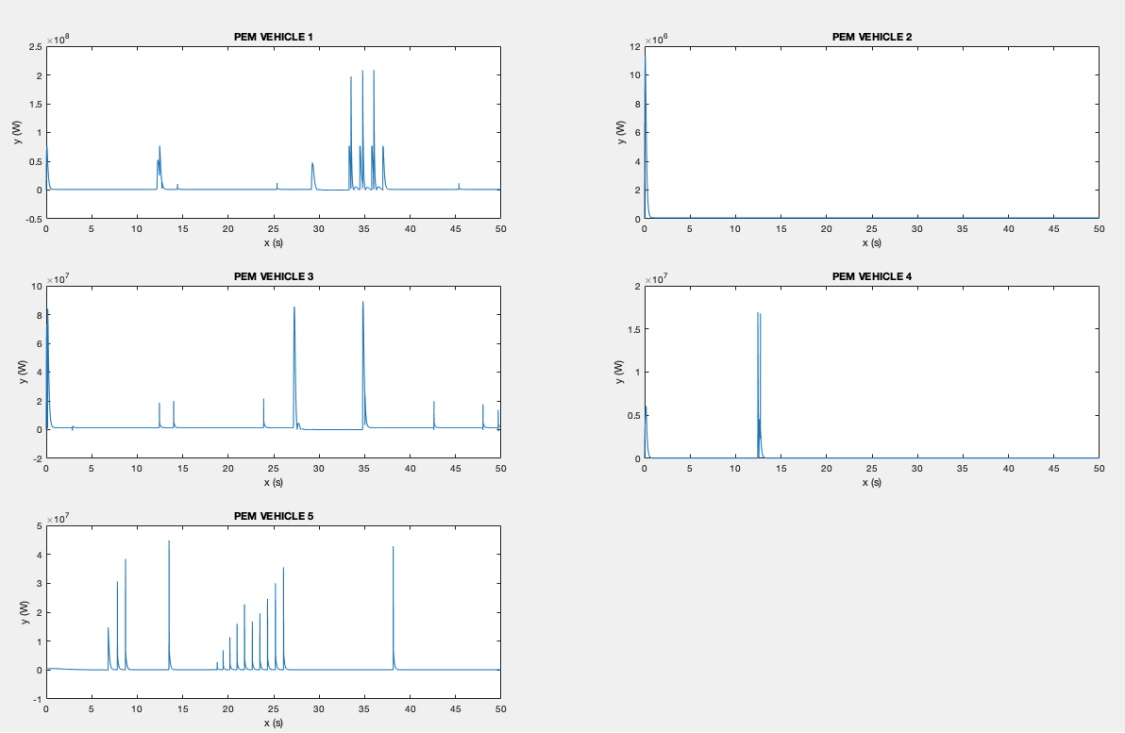


Figure. 17. PEM simulation result.